Tomáš Meravý Murárik
ais:127232

Zadanie 2a – klasifikácia

Riešený problém

- Postupné pridávanie bodov do 2D priestoru a ich následné klasifikovaie pomocou KNN algoritmu na základe bodov , ktoré sa na ploche už nachádzajú.

Opis riešenia

Riešenie pozostáva z 2 hlavných častí . Bežiaca časť a trieda Knn.

Bežiaca časť je kód, ktorý by normálne bol za podmienkou if __name__ == "__main__": takže všetky for-loopy a funkcie ktoré sú v nich vyvolávané . Bežiaca časť na začiatku vytvorý 3 listy (x,y,classes) kde sú uchované informácie o preddefinovaných bodoch a ich klasifikácia. (farby sú reprezentované ako čísla a na konci premenené naspäť)

```
x = np.empty(NUM_OF_POINTS + len(red[0]) * 4, dtype=np.int64)
y = np.empty(NUM_OF_POINTS + len(red[0]) * 4, dtype=np.int64)
classes = np.empty(NUM_OF_POINTS + len(red[0]) * 4, dtype=np.int64)
```

Hneď na to sa vytvoria 4 objekty triedy Knn kde každý objekt má iný počet klasifikovaných bodov.

Následne sa začnú generovať náhodné body a tie sa pridávajú do týchto objektov kde si ich už každý z nich sám klasifikuje podľa svojich parametrov.

```
for i in range(5, počet_bodov):
    for color in [farby]:
        point = gen_random_point(color)
        while point in points: #ak point nieje v points tak vytvoríme ďalší
            point = gen_random_point(*color[2])
        x[index] = point[0]
        y[index] = point[1]
        classes[index] = color[1]
```

Tomáš Meravý Murárik
ais:127232

```
        array1.add_point(point[0], point[1], index)
        array3.add_point(point[0], point[1], index)
        array7.add_point(point[0], point[1], index)
        array15.add_point(point[0], point[1], index)
```

Po vygenerovaní týchto bodov sa začne ich evaluácia a následné vykreslenie grafov.

```
accuracy_1 = evaluate_accuracy(array1)
accuracy_3 = evaluate_accuracy(array3)
accuracy_7 = evaluate_accuracy(array7)
accuracy_15 = evaluate_accuracy(array15)

print(f"k=1: {accuracy_1:.2f}%")
print(f"k=3: {accuracy_3:.2f}%")
print(f"k=7: {accuracy_7:.2f}%")
print(f"k=15: {accuracy_15:.2f}%")
```

Trieda Knn je hlavná časť tohoto zadania , ktorá vykonáva všetky dôležité výpočty. Ako argumenty pri inicializovaní berie:

1)nearest_num : premenná , ktorá reprezentuje , koľko najbližších bodov má brať do úvahy pri výpočtoch.
2)x : premenná , ktorá reprezentuje Xové pozície všetkých predošlích bodov.
3)y : premenná , ktorá reprezentuje Yové pozície všetkých predošlích bodov.
4)classes : premenná reprezentujúca farby daných bodov.
Trieda Knn má 2 hlavné funkcie add_point() a classify().
1)add_point() pridá bod do svojej kópie bodov a vyvolá funkciu classify
2)classsify() je funkcia na klasifikovanie bodov pomocou Euklidovskej metriky.

Začne tým že si urobí svoju kópiu bodov a classes iba od 0 po limited_index . Následne sa vykoná np.sqrt , ktorá vráti list vzdialeností od nášho bodu . Vzdialenosti sú potom zoradené od najmenšej po najväčšiu a z nich sa vyberie n najmenších a ich farby sú spočítané a returnne sa najčastejšia farba.

```
    def classify(self, x, y, limit_index):
        limited_x = self.x[:limit_index]
        limited_y = self.y[:limit_index]
```

Tomáš Meravý Murárik
ais:127232

```
    limited_classes = self.c[:limit_index]
    distances = np.sqrt((limited_x - x) ** 2 + (limited_y - y) ** 2)
    nearest_indices = np.argsort(distances)[: self.nn]
    nearest_classes = limited_classes[nearest_indices]
    most_common_class = Counter(nearest_classes).most_common(1)[0][0]
    return most_common_class
```

reprezentáciu údajov problému

údaje sú reprezentované zásadne ako čísla.
x= numpy array s x-ovými pozíciami bodov
y= numpy array s y-ovými pozíciami bodov
classes= numpy array s farbami , ktoré sú zakódované ako čísla
array1,array3,array7,array15 = sú objekty triedy Knn ktoré sa používajú na klasifikovanie bodov každá array má ale svoje vlastné x a y kde sa nachádza jú jej vlastné kópie bodov a v c sa nachádzajú ich farby.

```
x = np.empty(NUM_OF_POINTS + len(red[0]) * 4, dtype=np.int64)
y = np.empty(NUM_OF_POINTS + len(red[0]) * 4, dtype=np.int64)
classes = np.empty(NUM_OF_POINTS + len(red[0]) * 4, dtype=np.int64)
array1 = Knn(1, x, y, classes)
array3 = Knn(3, x, y, classes)
array7 = Knn(7, x, y, classes)
array15 = Knn(15, x, y, classes)
```

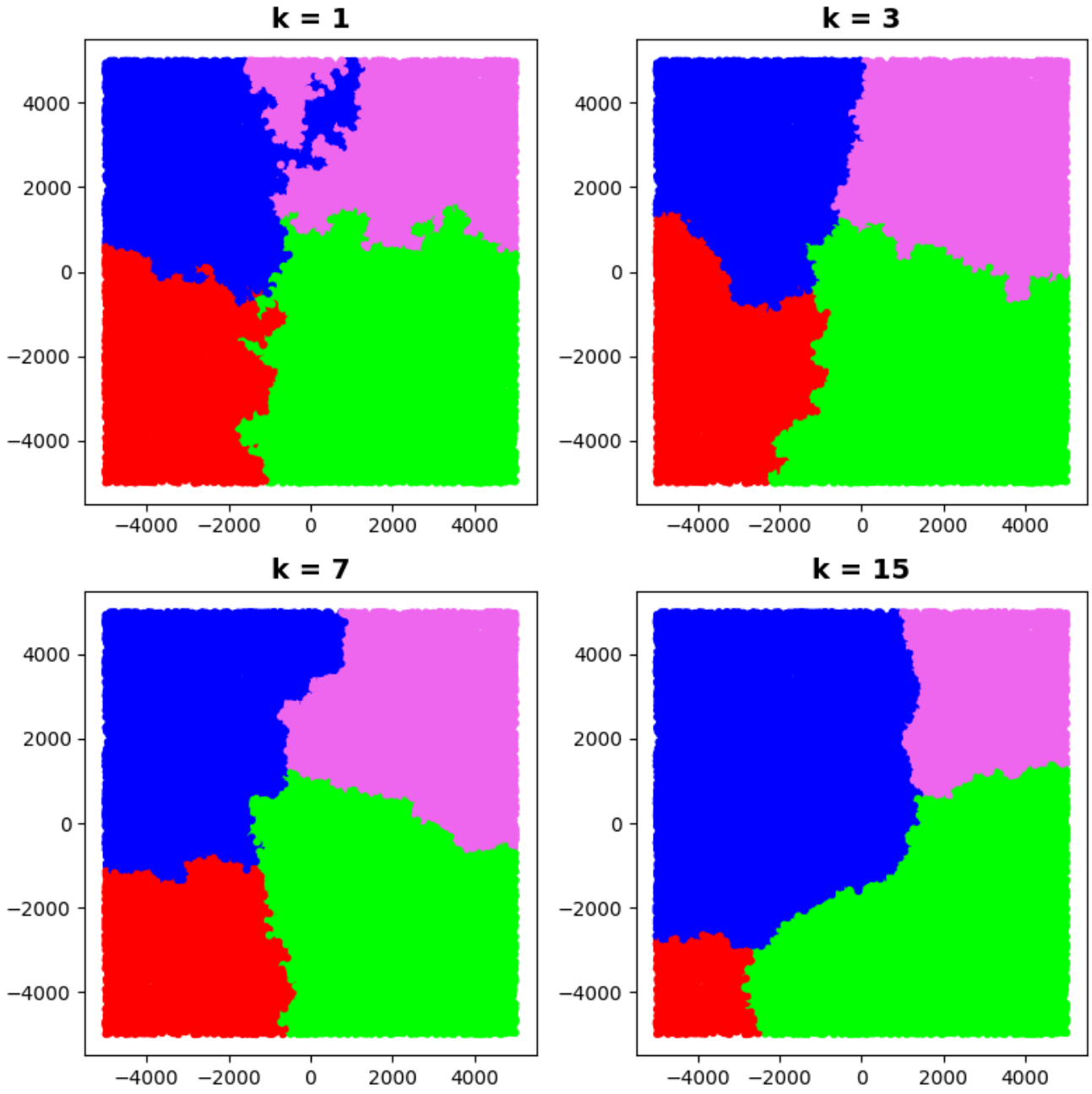
Spôsob testovania a výsledky experimentov, ladenia programu

Ladenie programu bol dlhý proces prepisovani všetkých hodnôt na numerickú reprezentáciu s cieľom rýchlejšieho behu programu.

zhodnotenie riešenia a dosiahnutých výsledkov

Riešenie iked nie optimálne je stále pomerne rýchle s časom približne 1 min na zbehnutie všetkých knn objektov a ich klasifikáciou. Program by sa dal zlepšiť rýchlejším algoritmom na klasifikáciu napr s použitím stromov alebo lepšiu reprezentáciou dát.

Tomáš Meravý Murárik
ais:127232




```
k=1: 72.90%
k=3: 71.47%
k=7: 73.68%
k=15: 55.74%
(env) [mary@Windows95 second]$ python3 main.py
k=1: 72.34%
k=3: 71.92%
k=7: 75.62%
k=15: 68.89%
(env) [mary@Windows95 second]$ python3 main.py
k=1: 71.61%
k=3: 75.39%
k=7: 73.26%
k=15: 60.68%
(env) [mary@Windows95 second]$ python3 main.py
k=1: 72.66%
k=3: 75.01%
k=7: 72.36%
k=15: 56.94%
(env) [mary@Windows95 second]$ python3 main.py
```